EmoContext: Contextual Emotion Detection in Text

Aditya Chetan IIIT Delhi
Brihi Joshi IIIT Delhi
Siddharth Yadav IIIT Delhi
{aditya16217, brihi16142, siddharth16268}@iiitd.ac.in

Abstract

This is the final report of our course project for the course CSE556 - Introduction to Natural Language Processing, taught by Dr. Tanmoy Chakraborty at IIIT Delhi. In this project, we tackle the problem of classifying the sentiment of an utterance in a dialogue into one of the three classes – “happy”, “sad” or “angry”. Given a user utterance with two turns of context, predict the sentiment of the user utterance from one of the three classes. We first explain the dataset and its technicalities which shape several design decisions later. We then explain the past work done to solve this problem, which includes our baseline that we have tried to beat. After that, we lay the ground work of our different architechtures. This problem statement is a part of the challenge tasks of the SemEval 2019 workshop.

1 Dataset

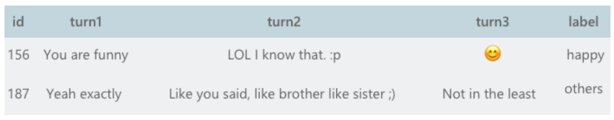
The training data is made *publicly available* by the SemEval contest organizers. The training data contains 30k records of labelled data which is divided into 3 5k long emotion categories - “happy”, “angry” or “sad”. The remaining 15k records belong to the “others” category. Apart from this labelled training data, the organizers have also made the unlabelled validation set.

The schema of the dataset is shown in Table 1. The dev set and the test set have a similar schema except for the “Label” column.

1.1 Dataset Details

The dataset has 5K records from each class in the training set and 15K records labelled as “others”. In the dev set, each of the three classes constitute 4% of the examples each whereas the rest of the examples belong to the “others” class.

An example of records in the dataset is given in Figure 1.1.



id	turn1	turn2	turn3	label
156	You are funny	LOL I know that. :p	😊	happy
187	Yeah exactly	Like you said, like brother like sister ;)	Not in the least	others

Figure 1: Snapshot of training data

1.2 Preliminary Observations

After a thorough review of the dataset, the following observations are made -

- Even though the length of the dataset seems to be long (most of the times, 30k records are sufficient to train a shallow network, here, the utterances are very small in length. Sometimes, they are also single words.
- The text in most of these utterances is common ‘sms-language’. This means that most words do not belong to the common english language dictionary. This can potentially create problems while creating distributed representations of words, i.e, word vectorisation seems to be a challenge.
- Apart from text, a large number of emojis are also present in the text. This can play both a positive and negative role. It can be effectively used to determine the emotion. However, parsing and handling them with text is a challenge.

2 Baselines

We use the work of the organizers (Gupta et. al., 2017) on a Sentiment-and-Semantics-Based Approach for emotion detection in textual conversations as a baseline for our method. Their model involved feeding concatenated GLoVe and

Col.	Field	Description
1	id	The unique ID of the record
2	Turn 1	Contains the first turn in the three turn conversation, written by User 1
3	Turn 2	Contains the second turn, which is a reply to the first turn in conversation and is written by User 2
4	Turn 3	Contains the third turn, which is a reply to the second turn in the conversation, which is written by User 1
5	Label	Contains the human judged label of Emotion of Turn 3 based on the conversation for the given training sample. It is always one of the four values - 'happy', 'sad, 'angry' and 'others'

Table 1: Schema of Training set.

SSWE embeddings to fully connected layer followed by a softmax layer to get the probabilities of the classes. On the shared dataset, their model achieves a **micro-averaged F1 score** of **0.5861**

Apart from this work, we could not find any other work that deals with this problem of detecting sentiment in contextual settings. Hence, for further steps, we will include all the models that we construct in the course of this contest as baselines.

3 Our Model

3.1 Model v1.0 - With DeepMoji

These are the set of experimentations that were tried using DeepMoji. The reason to use DeepMoji were manifold –

- They incorporated for the use of emojis in the utterances, which did not lead to the loss of information about emotions from the utterances.
- They inherently are a classification algorithm, which can classify a dialogue into a rich set of 64 categories, which can then be easily condensed into the 4 emotions that we need to categorise into.

3.1.1 Preprocessing – DeepMoji

We use DeepMoji (Felbo et. al., 2017) to convert the dialogues into a rich vectorised representation, that contains information about the sentiment/emotion of the dialogue, taking into account the Emojis in the text and the context of the dialogues.

DeepMoji essentially classifies the dialogue into one of 64 emojis. We use the second-to-last layer of their pre-trained model, a 2304 dimension vector as our embedding of the input dialogue. The reason we do so is on the assumption that this vector will encapsulate more information about the emotion in the dialogue, since it is being trained to match to appropriate emojis. This is what will go into out actual deep learning model.

3.1.2 The Model

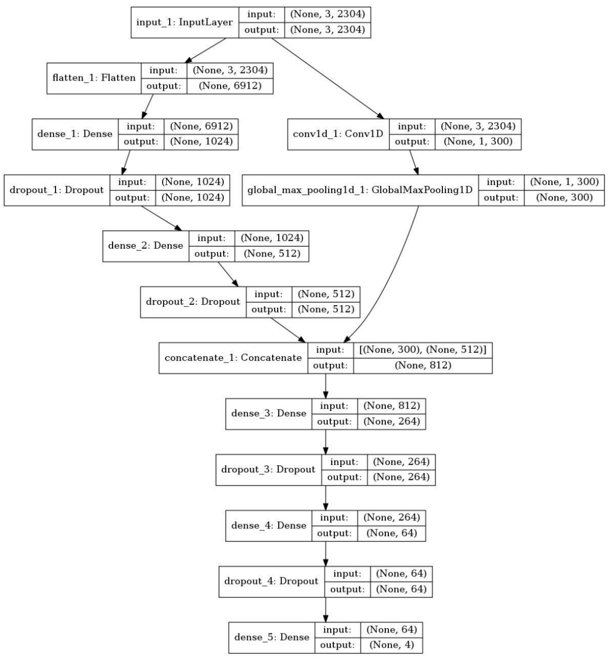
Our model thus takes in three vectors of dimension 2304. On one branch, these vectors are concatenated to each other forming a 6912 dimension vector. This is then fed to a 2-layered fully connected neural network, only to give out a 512 dimension vector. On the second branch, the three vectors are stacked up and fed to a 1D CNN with Global Max Pooling and 300 filters, to give a 300 dimension vector. The outputs of the two branches are then concatenated and fed into two more fully connected layers. We have used Dropout(with value of 0.2 to 0.5) after every dense layer. The final layer is a softmax layer for classification. A pictorial representation of the model in shown in figure 6.

4 Model v2.0 - With ELMo

In this set of experiments, instead of DeepMoji, we use ELMo embeddings. Several variations of the network afterwards is then tested.

4.1 ELMo Embeddings

ELMo is a deep contextualized word representation that models both complex characteristics of word use (e.g., syntax and semantics), and how these uses vary across linguistic contexts (i.e., to model polysemy). These word vectors are learned functions of the internal states of a deep bidirectional language model (biLM), which is pretrained on a large text corpus.



input_1: InputLayer | input: (None, 3, 2304) | output: (None, 3, 2304)
flatten_1: Flatten | input: (None, 3, 2304) | output: (None, 6912)
dense_1: Dense | input: (None, 6912) | output: (None, 1024)
conv1d_1: Conv1D | input: (None, 3, 2304) | output: (None, 1, 300)
dropout_1: Dropout | input: (None, 1024) | output: (None, 1024)
global_max_pooling1d_1: GlobalMaxPooling1D | input: (None, 1, 300) | output: (None, 300)
dense_2: Dense | input: (None, 1024) | output: (None, 512)
dropout_2: Dropout | input: (None, 512) | output: (None, 512)
concatenate_1: Concatenate | input: [(None, 300), (None, 512)] | output: (None, 812)
dense_3: Dense | input: (None, 812) | output: (None, 264)
dropout_3: Dropout | input: (None, 264) | output: (None, 264)
dense_4: Dense | input: (None, 264) | output: (None, 64)
dropout_4: Dropout | input: (None, 64) | output: (None, 64)
dense_5: Dense | input: (None, 64) | output: (None, 4)

Figure 2: Pictorial Representation of our model v1.0

4.2 Preprocessing

ELMo, unlike word2vec or doc2vec, is a character level embedding that can detect context from a very minute (context level). Unlike Word2vec, it can only generate embeddings for words that are there in its vocabulary. On the other hand, ELMo, generating character level embeddings can always generate embeddings for all words. Thus, they play a very important role in creating embeddings for words which belong to the ‘sms-language’.

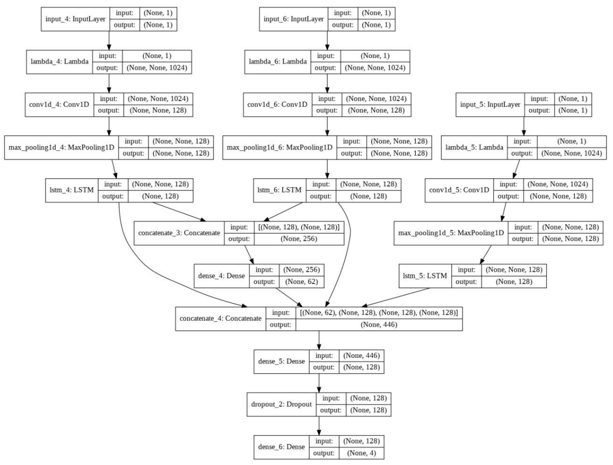
After using character level embeddings, ELMo uses them to generate word-level embeddings.

4.3 Using ELMo to generate one embedding for a set of utterances

After word level embeddings have been generated, the following steps are followed to create a single embedding a set of utterances -

- The 2-D word level embeddings are concatenated.
- 1-D convolution is used on top of them.
- Every filter of the convolution is then ordered and sent to the LSTM.
- The output of the LSTM is what we consider as the embedding of the one of the sentences of the whole utterance.
- Finally, for all three sentence embeddings, we concatenate them to create the embedding of the overall utterance.
- This overall utterance is then sent to the other higher layers, the different variations of which we present in the subsequent sections.

A figure describing this procedure, along with the dimensions of each vector is shown here -



input_4: InputLayer | input: (None, 1) | output: (None, 1)
input_6: InputLayer | input: (None, 1) | output: (None, 1)
lambda_4: Lambda | input: (None, 1) | output: (None, None, 1024)
lambda_6: Lambda | input: (None, 1) | output: (None, None, 1024)
conv1d_4: Conv1D | input: (None, None, 1024) | output: (None, None, 128)
conv1d_6: Conv1D | input: (None, None, 1024) | output: (None, None, 128)
input_5: InputLayer | input: (None, 1) | output: (None, 1)
max_pooling1d_4: MaxPooling1D | input: (None, None, 128) | output: (None, None, 128)
max_pooling1d_6: MaxPooling1D | input: (None, None, 128) | output: (None, None, 128)
lambda_5: Lambda | input: (None, 1) | output: (None, None, 1024)
lstm_4: LSTM | input: (None, None, 128) | output: (None, 128)
lstm_6: LSTM | input: (None, None, 128) | output: (None, 128)
conv1d_5: Conv1D | input: (None, None, 1024) | output: (None, None, 128)
concatenate_3: Concatenate | input: [(None, 128), (None, 128)] | output: (None, 256)
max_pooling1d_5: MaxPooling1D | input: (None, None, 128) | output: (None, None, 128)
dense_4: Dense | input: (None, 256) | output: (None, 62)
lstm_5: LSTM | input: (None, None, 128) | output: (None, 128)
concatenate_4: Concatenate | input: [(None, 62), (None, 128), (None, 128), (None, 128)] | output: (None, 446)
dense_5: Dense | input: (None, 446) | output: (None, 128)
dropout_2: Dropout | input: (None, 128) | output: (None, 128)
dense_6: Dense | input: (None, 128) | output: (None, 4)

Figure 3: Pictorial Representation of the ELMo System

4.4 ELMo + CNN + LSTM

The CNN+LSTM Model is displayed in Figure 3. This particular model has given us an increased F1-Score of **0.627**.

This model is compared with two other models -

- Only CNN - **0.621**
- Only LSTM - **0.634**

These two other models can be visualised by removing and addition of components in Figure 3.

However, as we observe, ELMo based models was not very advantageous for us. We suspect that the reasons for this were as follows:

- The embeddings for each sentence generated by ELMo have a very high dimension.
- On an already complicated, yet small dataset, taking such complex features increases the VC dimension of the problem, and thus leads to overfitting.

5 Model v3.0 - Data Augmentation

After doing some diagnostic tests, we figured out that all our models had been overfitting. This was because:

- With all possible model tests that we did, the training accuracy was always much more than the test accuracy.
- Even though in normal cases, the training accuracy is more than test accuracy, the overfit can be detected by the difference in the accuracies of both. In our case, the training accuracies were shooting upto 0.88 - 0.90 whereas the test accuracies only remained oscillating around 0.63.

This is the reason why we realized that probably, the less amount of data that we have is leading to the overfit. Thus, we tried some data augmentation techniques that would increase our data size and help in getting a better fit.

5.1 Augmentation Technique 1

The first technique that we use is called **word switch**. We switch some non-padding words in a sentence with each other to generate new sentences with meanings close to the original sentences.

5.2 Augmentation Technique 2

The second technique that we use is called **word removal**. In this technique, we randomly remove a non-padded utterance from a sentence and append the resultant sentence to the training set.

Using these two techniques, we managed to increase the training set data size to 53000. Still we did not see much better results as the F1 score still remained around 0.66.

6 Model v4.0 - `vocabBoost` DeepMoji + Data augmentation

Our final experiment involved DeepMoji with an increased vocabulary size of 3000 along with an augmented Training Dataset. This had a two-fold advantage:

- DeepMoji has extremely lower VC dimension as compared to ELMo, and thus, a very lower chance of overfitting.
- Also, an augmented dataset also helps in preventing overfitting.

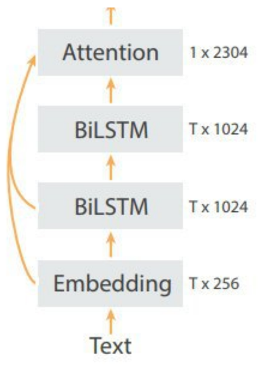


Figure 4: DeepMoji Architecture

7 Experimentation and Results

We currently do not have access to the annotated test set and dev set. So, for the purpose of training, we split our training set provided by the organizers into train-dev sets (2:1) and trained and evaluated the model on this distribution. We trained it for 10 epochs. Further training doesn't change the validation accuracy and decreases the test f1 score(in other words it starts overfitting). The results are shown in Table 2.

Statistic	**Micro**	**Macro**
Precision	0.8863	0.8866
Recall	0.8316	0.8579
F1-Score	0.8763	0.8720

Table 2: Schema of Training set.

We trained on the entire dataset to get a micro F1-score of **0.678351**. The hosts' baseline had a micro F1-score of **0.5**

7.1 Additional Information

We are currently in the top 65 teams worldwide and the first team from IIIT Delhi.

References

Gupta et. al. 2017. *Computing Research Repository.* https://arxiv.org/abs/1707.06996

Felbo, Bjarke and Mislove, Alan and Søgaard, Anders and Rahwan, Iyad and Lehmann, Sune. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. 2017 *Conference on Empirical Methods in Natural Language Processing (EMNLP).*